**Описание приложения "CFA APP"**

Онлайн-финансовое приложение "CFA APP" представляет собой платформу, позволяющую пользователям управлять своими финансами, отслеживать активы, проводить финансовые операции, а также отслеживать историю транзакций. Пользователи имеют возможность просматривать текущий баланс, инвестировать в различные финансовые инструменты на бирже, а также пополнять свой счет.

**Технический стек:**

Frontend: Приложение построено на базе React, который использует компонентную архитектуру для удобного разделения интерфейса на независимые блоки. Для маршрутизации между различными разделами используется React Router. Для визуализации графиков цен используется библиотека Chart.js.

Backend: Взаимодействие с серверной частью осуществляется с помощью HTTP запросов. На сервере может использоваться Express.js, чтобы обрабатывать эти запросы, а также обеспечивать аутентификацию и авторизацию пользователей. Для хранения данных о пользователях, активах и транзакциях может быть использована база данных, например, MongoDB.

**Структура проекта:**

1. src/components: Эта директория содержит React компоненты, которые используются для построения пользовательского интерфейса. Каждый компонент отвечает за отображение определенной части приложения, например, Header, StockMarket, MyAssetsPage и т.д.
2. src/images: Здесь хранятся изображения, используемые в приложении.
3. src/styles: В этой директории содержатся стили CSS для оформления компонентов.
4. src/App.js: Основной файл приложения, где настраиваются маршруты и рендерятся компоненты.
5. src/index.js: Входная точка приложения, где происходит рендеринг приложения в DOM.
6. public: Эта директория содержит статические файлы, такие как index.html.
7. server: Если вы создаете серверную часть приложения, здесь могут находиться файлы, связанные с сервером.

**Функциональные возможности:**

1. Аутентификация и Регистрация: Пользователи могут создавать аккаунты и авторизовываться, чтобы получить доступ к своему личному кабинету.
2. Баланс и Транзакции: Пользователи могут видеть свой текущий баланс, а также просматривать историю своих финансовых транзакций.
3. Биржа и Инвестирование: Пользователи могут просматривать доступные финансовые инструменты на бирже, а также инвестировать в них.
4. Управление Активами: Пользователи могут управлять своими активами, просматривая их список, текущую стоимость и количество.
5. Пополнение Счета: Пользователи имеют возможность пополнять свой счет через различные методы оплаты.

**Схема взаимодействия:**

1. Пользователь открывает веб-приложение в браузере.
2. Приложение загружает компоненты, маршруты и стили.
3. Пользователь аутентифицируется или регистрируется, получая доступ к своему аккаунту.
4. Пользователь взаимодействует с разделами приложения: просматривает активы, инвестиции, транзакции и др.
5. Приложение отправляет HTTP запросы на серверную часть для получения или обновления данных.
6. Сервер обрабатывает запросы, работает с базой данных и возвращает данные обратно в приложение.
7. Пользователь видит обновленные данные на экране.

**API:**

Реализован на python с помощью FastAPI, база данных – postgres.

API соединена с базой данных приложения благодаря docker-compose.

Команда для запуска

cd Finodays/DigiAsset && docker-compose up -d

API доступен по адресу 0.0.0.0:8000, а документация по нему по адресу 0.0.0.0:8000/docs#/

Спецификация для API:

1. Метод: GET

URL: /

Описание: Главная страница

Возвращаемое значение: Словарь с приветственным сообщением

2. Метод: GET

URL: /products

Описание: Получение списка продуктов

Параметры:

- category (опциональный): категория продуктов

Возвращаемое значение: JSON-ответ со списком продуктов

3. Метод: GET

URL: /products/{id}

Описание: Получение продукта по id

Параметры:

- id: id продукта

Возвращаемое значение: JSON-ответ с продуктом

4. Метод: GET

URL: /orders/{user\_id}

Описание: Получение заказов пользователя по id

Параметры:

- user\_id: id пользователя

Возвращаемое значение: JSON-ответ со списком заказов

5. Метод: POST

URL: /orders

Описание: Создание заказа

Параметры:

- order\_list: список заказов

Возвращаемое значение: JSON-ответ с подтверждением создания заказа

6. Метод: GET

URL: /orders/{id}

Описание: Получение заказа по id

Параметры:

- id: id заказа

Возвращаемое значение: JSON-ответ с заказом

7. Метод: PUT

URL: /orders/{id}/cancel

Описание: Отмена заказа

Параметры:

- id: id заказа

Возвращаемое значение: JSON-ответ с информацией об отмене заказа

8. Метод: PUT

URL: /orders/{id}/confirm\_payment

Описание: Подтверждение оплаты заказа

Параметры:

- id: id заказа

Возвращаемое значение: JSON-ответ с информацией о подтверждении оплаты заказа